Generative Goal-driven User Simulation for Dialog Management

Generative Goal-Driven User Simulation for Dialog Management

Aciel Eshky
ILCC
School of Informatics
University of Edinburgh
a.eshky@sms.ed.ac.uk

Ben Allison
ILCC
School of Informatics
University of Edinburgh

Mark Steedman
ILCC
School of Informatics
University of Edinburgh
{ballison, steedman}@inf.ed.ac.uk

Abstract

User simulation is frequently used to train statistical dialog managers for task-oriented domains. At present, goal-driven simulators (those that have a persistent notion of what they wish to achieve in the dialog) require some task-specific engineering, making them impossible to evaluate intrinsically. Instead, they have been evaluated extrinsically by means of the dialog managers they are intended to train, leading to circularity of argument. In this paper, we propose the first fully generative goal-driven simulator that is fully induced from data, without hand-crafting or goal annotation. Our goals are latent, and take the form of topics in a topic model, clustering together semantically equivalent and phonetically confusable strings, implicitly modelling synonymy and speech recognition noise. We evaluate on two standard dialog resources, the Communicator and Let's Go datasets, and demonstrate that our model has substantially better fit to held out data than competing approaches. We also show that features derived from our model allow significantly greater improvement over a baseline at distinguishing real from randomly permuted dialogs.

1 Introduction

Automatically simulating user behaviour in human-machine dialogs has become vital for training statistical dialog managers in task-oriented domains. These managers are often trained with some variant of reinforcement learning (Sutton and Barto, 1998), where optimal behaviour is sought or learnt through the exploration of the space of possible dialogs. Although learning by interacting with human subjects is a possibility (Gašić et al., 2011), it has been argued that user simulation avoids the expensive, labour intensive, and error-prone experience of exposing real humans to fledgling dialog systems (Eckert et al., 1997).

Training effective dialog managers should benefit from exposure to properties exhibited by real users. Table 1 shows an example dialog in a domain such as we consider, where the objective is to simulate at the semantic level. In such task oriented domains, the user has a *goal* (in this case, to book a flight from New York to Osaka), and the machine is tasked with fulfilling it. Notice that the user is consistent with this goal throughout the dialog, in that they do not provide contradictory information (although an ASR error is present), but that every mention of their destination city uses a different string. This motivates our first desideratum: that simulation be consistent over the course of a dialog. Furthermore, one can imagine users not always responding identically in identical situations: we thus additionally require *variability*. In this paper we demonstrate a fully generative, latent variable probability model exhibiting both of these properties.

Thus far, consistent simulators have been partially deterministic and have required some hand-engineering. As a result, it has only been possible to evaluate them extrinsically using dialog managers. This is circular because we need simulators to train managers, but need managers to evaluate simulators. The issue is that judgements of quality of each depend on the specifics of the other and that a proper evaluation of one depends on the correct functioning of the other. Furthermore, there is little reason to assume that because a simulator performs well with a certain dialog manager, it would perform similarly

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 71–81, Jeju Island, Korea, 12–14 July 2012.

Speech	Semantic Representation
M: Hello, How Can I help?	M: GREETING M: META_REQUEST_INFO
U: A trip from New York City to Osaka, please.	U: PROVIDE *orig_city* **New York City** U: PROVIDE *dest_city* **Salt Lake City**
M: Leaving from New York City to Salt Lake City. What day would you like to travel?	M: IMPLICIT_CONFIRM *orig_dest_city* M: REQUEST *depart_date*
U: No, no. Leaving from New York to Osaka in Japan.	U: NO_ANSWER *null* **no** U: PROVIDE *orig_city* **New York** U: PROVIDE *dest_city* **Osaka Japan**
M: Leaving from New York to Osaka Japan, correct?	M: EXPLICIT_CONFIRM *orig_city* M: EXPLICIT_CONFIRM *dest_city*
U: Yes.	U: YES_ANSWER *null* **yes**

Table 1: An example of a dialog in speech and its semantic equivalent. M and U denote machine and user utterances respectively. Note how a single speech utterance is split by the semantic parser into multiple logical utterances, each of which is broken down to an ACT, *slot*, and **value**. We consider resources where gold standard transcriptions are not available; thus there will be speech recognition noise, e.g. **Osaka** rendered as **Salt Lake City**, something our model is able to capture.

well with other managers. In contrast, a probabilistic formulation such as we propose allows us to evaluate our models intrinsically using standard machine learning metrics, and without reference to a *specific* manager, thus breaking the circularity, and guarding against such experimental biases.

We demonstrate the efficacy of our model on two tasks, and compare it to two other approaches. Firstly we use a standard bigram model as conceived by Eckert et al. (1997) and Levin and Pieraccini (2000); secondly we compare to a probabilistic goal-based simulator where the goals are string literals, as envisaged by Scheffler and Young (2002) and Schatzmann et al. (2007b). We demonstrate substantial improvement over these models in terms of predicting heldout data on two standard dialog resources: DARPA Communicator (Levin et al., 2000; Georgila et al., 2005b) and Let's Go (Black and Eskenazi, 2009).

2 Related Work

2.1 Related Work on User Simulation

User simulation as a stochastic process was first envisioned by Eckert et al. (1997): their *Bigram* model conditions user utterances exclusively on the preceding machine utterance. This was extended by Levin and Pieraccini (2000), who manually restrict the model to estimating "sensible" pairs of user and machine utterances by assigning all others probability zero.

Bigram models ensure that a locally sensible response to a machine utterance is provided by the simulator; however, they do not ensure that it provides responses consistent with one another throughout the dialog. Several approaches have attempted to overcome this problem. Pietquin (2004), for example, explicitly models a user goal as a set of slot-value pairs randomly generated once per dialog. He then hand selects parameters to ensure that the user's actions are in accordance with their goal.

Jung et al. (2009) use large amounts of dialog state annotations (e.g. what information has been provided so far) to learn Conditional Random Fields over the user utterances, and assume that those features ensure user consistency. Georgila et al. (2005a) instead consider only act-slot pairs, and thus inconsistency is not a factor.

Scheffler and Young (2002) simulate user behaviour by introducing rules for actions that depend on the user goal, and probabilistic modelling for actions that are not goal-dependent. They then map out a decision network that determines user actions at every node prior to the start of the dialog. Agenda-based user simulation, another approach from the literature, assumes a probability distribution over the

user goal which is either induced from data (Schatzmann et al., 2007b), or is manually set when no data is available (Schatzmann et al., 2007a). An agenda, which is a stack-like structure of utterances to be produced given the goal, is then devised deterministically. Keizer et al. (2010) combine the decision network with the agenda and goal to allow for some variability for some actions. These models ensure consistency but restrict the variability in user behaviour that can be accommodated. Furthermore, because these approaches do not define a complete probability distribution over user behaviour, they restrict possibilities for their evaluation, a point to which we now turn.

2.2 Related Work on Simulator Evaluation

No standardised metric of evaluation has been established for user simulators largely because they have been so inextricably linked to dialog managers. The most popular method of evaluation relies on generating synthetic dialogs through the interaction of the user simulator with some dialog manager. Schatzmann et al. (2005) hand-craft a simple deterministic dialog manager based on finite automata, and compute similarity measures between these synthetically produced dialogs and real dialogs. Georgila et al. (2006) use a scoring function to evaluate synthetic dialogs using accuracy, precision, recall, and perplexity, while Schatzmann et al. (2007b) rely on dialog completion rates. Williams (2008) use a Cramer–von Mises test, a hypothesis test to determine whether simulated and real dialogs are significantly different, while Janarthanam and Lemon (2009) use Kullback Leibler Divergence between the empirical distributions over acts in real and simulated dialogs. Singh et al. (2000) and Ai and Litman (2008) judge the consistency of human quality ranked synthetic dialogs generated by different simulators interacting with the IT-SPOKE dialog system.

Schatzmann et al. (2007b) use a simulator to train a statistical dialog manager and then evaluate the learned policy. Because this only indirectly evaluates the simulator, it is inappropriate as a sole measure of quality.

There has been far less evaluation of simulators without a dialog manager. The main approach is to compute precision and recall on an utterance basis, which is intended to measure the similarity between real user responses in the corpora and simulated user responses produced under similar circumstances (Schatzmann et al., 2005; Georgila et al., 2006). However, this is a harsh evaluation as it assumes a correct or "best" answer, and penalises valid variability in user behaviour.

3 Dialog as a Statistical Process

We consider a dialog to be a series of *turns*, comprised of multiple *utterances*. Each Utterance consists of an ACT, a *slot*, and a **value**, as shown in Table 1. Dialogs proceed by the user and the machine alternating turns. Because the dialogs are of mixed initiative, there is no restriction on the number of contiguous machine or user utterances.

Our aim is to model the user, and are interested in the conditional distribution of the user utterances given the dialog up to that point. In other words, we are interested in the distribution $p\left(u_i|d_1 \dots d_{i-1}\right)$, where d_n is either a machine utterance m_n or a user utterance u_n.

4 Models of Users in Dialogs

This section describes several models of increasing complexity: a Bigram model, which serves as a baseline; an upper-bound on String-Goal models, which we design to mimic the behaviour of previous goal-based approaches, but with a probabilistic formulation; and finally our approach, the Topic-Goal model.

4.1 Bigram Model

The simplest model we define over dialogs is the bigram model of Eckert et al. (1997):

$$p\left(u_i|\mathbf{m}\right) = p\left(u_i|m_{i-1}\right) \quad (1)$$

$$p\left(\mathbf{u}|\mathbf{m}\right) = \prod_i p\left(u_i|\mathbf{m}\right) \quad (2)$$

The probability of each user utterance u_i (the complete $\{$ACT, *slot*, **value**$\}$ triple) is dependent only on the machine utterance immediately preceding it (the slight abuse of notation m_{i-1} here does not mean the utterance at $i-1$ in the machine utterance list, but the utterance immediately preceding the i-th), and utterances in the dialog are conditionally independent of

one another. (Georgila et al. (2006) found no benefit from increasing the Markov horizon). Since each utterance is generated independently of others in the dialog with the same context, there is no enforced consistency between utterances.

Since we require a distribution over all possible utterances, assigning non–zero probability to cases outside of the training data, our bigram model is interpolated with a unigram model, which itself is interpolated with a smoothing model which assumes independence between the act, slot, and value elements of the utterance. Interpolation weights are set to maximise probability of a development set of dialogs. Each sub-model uses the maximum likelihood estimator (the relative frequency of the utterance), and unseen machine utterances place full weight on the unigram/smoothed model (ignoring the bigram probability since it has no meaning if m_{i-1} is unobserved). We label this model the Bigram model in subsequent experiments.

4.2 Goal-Based Models

One way to ensure consistency and more realistic behaviour is to have a *goal* for the user in the dialog, which corresponds to values for slots required in the problem. For instance, they might be the origin and destination cities in a flight booking domain. In standard machine learning terms, the goal becomes a latent variable g in a probability model. We can then define a distribution over utterances as:

$$p(u_i|\mathbf{m}, g) = p(u_i|m_{i-1}, g) \quad (3)$$

$$p(\mathbf{u}|\mathbf{m}) = \sum_g p(g) \prod_i p(u_i|m_{i-1}, g) \quad (4)$$

4.3 An Upper-Bound on String-Goal Models

The simplest variant of g has string values for each of the slots the user is required to provide in order for the dialog to succeed. Thus we may have:

$$g = [\textit{orig\_city}\text{: } \textbf{New York}; \textit{dest\_city}\text{: } \textbf{Osaka}]$$

as presented in Schatzmann et al. (2005) and Schatzmann et al. (2007b). However, in these simulators, while the goal is probabilistic, there is no distribution over utterances given the goal because utterances are assembled deterministically from a series of rule applications. There is also no marginalisation over the goal as in (4) above.

The issue with a model of user goals as strings in this fashion is that users describe the same values in multiple ways (**Osaka Japan**, **Osaka**), and speech recognition errors corrupt consistent user input (**Osaka** mis-recognised as **Salt Lake City**). Users also might legitimately switch their goals mid-dialog. Inference in the model would have allow for these possibilities: we would have to marginalise over all possible goal switches.

For the sake of comparison, we compute an upper-bound on string-goal models, which gives a flavour for how such models would perform *optimistically*. The upper-bound assigns probability to dialogs as follows: for each utterance u_i if the corresponding value v_i has been seen before in the dialog, the probability used for that utterance is just $p(a_i, s_i|m_{i-1})$, that is, the probability of the act a_i and slot s_i only; there is no penalty for repetition of the value. If the value is unseen in the dialog, we use the full probability of the utterance from the bigram model as described above. This is optimistic because there is no penalty for repeated goal changes besides that imposed by the bigram model itself, and no penalty is imposed for choosing between previously sampled goals as would be necessary in a probability model.

Any string-based model necessarily assigns lower probabilities to data than the upper bound, because it would penalise goal changes (in a probabilistic sense; that is, there would be a term to reflect the probability of some new goal given the old) to allow for the discrepancy in values present in dialogs. In contrast, our upper bound does not include such a term. Furthermore, once multiple goal values had been uttered in the dialog, we would have to sample one to use for the next utterance, which would again incur some cost: again, we do not have such a cost in our upper bound.

We could in theory use an external model of noise to account for these value discrepancies (and the ASR errors we model in the next section). However, this would further decrease the probability, as some probability mass currently assigned to the heldout data would have to be reserved for the possibility of string renderings other than those we observe.

It bears reiterating that our upper bound on string-goals is not a generative model: however, it allows us to assign probabilities to unseen data (albeit optimistically), and thus provides us with a point of

comparison. Although not technically a model, we refer to this as the String-Goal model for the remainder of the paper.

4.4 Topic-Goal Model

To motivate our proposal, consider that over the course of a dialog one could look at the set of all values used for some slot, for example the destination city, as a count vector:

$$v_{\text{dest\_city}} = \textbf{Salt Lake}{:}1;\ \textbf{Osaka}{:}2;\ \textbf{Osaka Japan}{:}1$$

The above vector may arise because the user actually wants to go to **Osaka**, but the destination is initially mis-recognised as **Salt Lake**, and the user finally disambiguates with the addition of the country. Such situations are common in the noisy dialog resources from which simulators are induced—however, any string-based goal will necessarily consider these different renderings to be different goals, and will require resampling or smoothing terms to deal with them.

Our approach instead treats the count vector as samples from a topic model; that is, a mixture over multinomial distributions. Whilst by far the most popular topic model is LDA (Blei et al., 2003), it provides too flexible a distribution over count vectors to be used with such small samples (we confirmed the poor suitability of this model in preliminary experiments). Instead we use the simpler Mixture-of-Multinomials model, where the latent topic is sampled once per dialog instead of once per value uttered. We describe below how parameters to this model are estimated, and focus for now on how the resulting model assigns probability to dialogs.

In this formulation, the latent goal for each slot, which was previously a string, now becomes an indicator for a topic in a topic model. Each topic can in theory generate any string (so the model is inherently smoothed), but most strings in most topics will have only the smoothing weight and most probability mass will be on a small number of highly correlated strings. We treat the slots as being independent of one another in the goal, and thus:

$$p(g) = \prod_s p\left(z_s\right) \tag{5}$$

Where z_s is the topic indicator for some slot s. If slot s has associated with it a count vector of values $\mathbf{v_s}$, each looking like the example above, then the distribution over the values used for each slot becomes:

$$p\left(v_s\right) = \sum_{z_s} p\left(z_s\right) p\left(v_s|z_s\right) \tag{6}$$

We then define a bigram-based Act model to describe the probabilities of the $\{$ACT, *slot*$\}$ pairs to which these values belong, so that:

$$p\left(\mathbf{u}|\mathbf{m}\right) = \prod_s p\left(z_s\right) \cdot \prod_i p\left(a_i, s_i|m_{i-1}\right) p\left(v_i|z_{s_i}\right) \tag{7}$$

In reality, some slots will not have corresponding values, or will be slots whose values are not appropriate to model in the above way. Dates and times, for example, have ordinal and structural relations between them, and a model which treats them as disconnected entities is inappropriate. For utterances defined over such slots we use a standard bigram model as in (1), and for appropriate utterances we use a topic-goal model as in (7). This constitutes the only domain knowledge necessary to adapt the model for a new resource. We refer to this model as the Topic-Goal model.

4.4.1 Topic Model Parameter Estimation

Our topic model is a Bayesian version of the Mixture-of-Multinomials model. Under this model, each dialog has associated with it a latent variable z_s for each slot s in the goal, which indicates which topic is used to draw the values for that slot. Conditioned on z, independent samples are drawn from the distribution over words to which that value of z corresponds—however, the effect in the marginal distribution over words is to strongly prefer sets which have co-occurred in training as these are assigned to the same topic.

Bayesian inference in mixture models has been described in detail in Neal (1991) and Griffiths and Steyvers (2004), so we give only a brief account here for our particular model. We take r appropriately-spaced samples from a Gibbs' sampler over the posterior mixture parameters θ, ϕ: θ are the word-topic parameters and ϕ are the mixture proportions. We assume a uniform Dirichlet prior on θ and ϕ, leading to Dirichlet posteriors which we integrate out in the predictive distribution over v using the standard Dirichlet integral. For each of our r samples we have

components z parameterised by γ_{rz} (the Dirichlet parameter for the z-th mixture component in the r-th sample) and α_{rzj} for each word j in the z-th topic for the r-th sample. The $\bullet$ notation indicates a sum over the corresponding index, i.e. $\gamma_{r\bullet} = \sum_z \gamma_{rz}$. Then:

$$p(v) = \frac{1}{|r|} \sum_r \sum_z \frac{\gamma_{rz}}{\gamma_{r\bullet}} p(v|\alpha_{rz}) \quad (8)$$

$$p(v|\alpha) = \frac{\Gamma(\alpha_\bullet)}{\Gamma(\alpha_\bullet + v_\bullet)} \prod_j \frac{\Gamma(v_j + \alpha_j)}{\Gamma(\alpha_j)} \quad (9)$$

This states that each of the r samples has topics z which are multinomial distributions with posteriors governed by parameters α_{rz}. For any of these topics, the distribution over v is as given in Equation (9) (we suppress the subscripting of α here for the different samples and topics, since this holds whatever its value). The final predictive probability given in Equation (8) averages over the samples r and the topics z (with topics weighted by their parameters γ_{rz}).

5 Experimental Setup

Our experiments use two standard corpora, the first of which is DARPA Communicator (DC), a flight booking domain collected between 2000-2001 through the interaction of real users with 10 different systems (Levin et al., 2000). It was later automatically annotated by Georgila et al. (2005b) to include semantic information. The second corpora is Let's Go (LG), years 2007, 2008, and 2009, distributed as part of the Spoken Dialog Challenge (Black and Eskenazi, 2009). Let's Go is a bus routing domain in Pittsburgh collected by having the general public interact with the CMU dialog system to find their way through the city. The dialogs in both corpora are of mixed-initiative, having a free number of contiguous system and user responses.

We preprocessed the corpora, converting Communicator XML-tagged files and Let's Go system log files into sequences of ACT, *slot*, and **value** utterances. Table 2 gives examples. We then divided the corpora into training, development and test sets as follows: Communicator contains 2285 dialogs in total, and Let's Go contains 17992, and in each case we selected 80% of dialogs at random for training, 10% for development, and 10% for testing.

DC:	PROVIDE_INFO *orig_city* **Boston**
LG:	INFORM *place* [departure_place **CMU**, arrival_place **airport**]

Table 2: Example utterances from the two corpora. Note how in addition to the value, the Let's Go utterances contain properties (departure_place and arrival_place).

Let's Go is a noisy corpus that contains far more speech recognition errors than Communicator. In addition, users tend to be more flexible with their bus routes than they are with their flight destinations, and so values are a lot more varied throughout the course of Let's Go dialogs than Communicator ones. Furthermore, Let's Go semantic parses contain ambiguity not present in Communicator; the parser fails to distinguish departure from arrival places over 90% of the time, and instead assigns them a generic Single_Place property. Our current model assumes the decisions made by the semantic parser are correct. In reality however, a better model would incorporate potential noise in the semantic parse in a joint model. We defer this more complex treatment for future work.

Free model parameters are set by a simple search on the development set, where the objective is likelihood—for the bigram model the parameters are the interpolation weights, and for the topic model we search for the number of topics and smoothing constant for the topic distributions. For Let's Go, since we can have multiple places provided in a single act, we treat each utterance as containing a set of values and build the count vector for the topic model as the union of these sets over the whole dialog. The slots over which the topic model is defined for Communicator are *dest_city* and *orig_city* (this takes into account PROVIDE and REPROVIDE acts). For Let's Go we derive the model over the three properties: single_place, arrival_place and departure_place, as opposed to the less informative slot *place*.

6 Evaluating the Simulators

We evaluate each of the models in terms of the probability they assign to the test data. This metric is more suitable than the precision and recall metrics which have been previously used, because it acknowledges that, rather than each user response being "correct" at the point which it is observed, there

Model	DC(A)	DC(P)	LG(A)	LG(P)
Topic	**252.78**	**860.2**	**113.45**	**1417.06**
String	270.09	1286.03	169.87	4578.23
Bigram	347.88	5979.53	223.23	10125.87
Act	9.56	5.2	2.77	2.34

Table 3: The mean per-utterance perplexity on heldout data. DC-A is all acts for Communicator, while DC-P is the calculated on PROVIDE acts alone (the acts on which our model is designed to improve prediction). LG-A and LG-P have the same meaning for Let's Go.

is a distribution over possible responses. Because the models we define are full probability models, we are able to compute this metric and do not need to use an arbitrarily selected dialog manager for evaluation.

The heldout probability metric should be understood as a means of comparing the relative viability of different models of the same data. Note that we are reporting the probability of unobserved data, rather than data from which the models were induced, and are thus measuring the generalisability of the models (in contrast, maximising the probability of the training data would simply encourage overfitting). The absolute numbers are hard to interpret, as there is no hard upper bound; while it may be appealing to think of an upper bound of 1, this is incorrect as it would imply that there was no variability in the data. However, it should be understood that assigning particular behaviour higher probability means that the model is more likely to exhibit it when run in simulation mode—and since the user behaviour in question has not been seen at training time, this measures the extent to which the models have generalised beyond the training data relative to one another.

We report the mean per-utterance log probability of unseen data, that is, the probability of the whole heldout corpus divided by the number of user utterances.

6.1 Results

Figure 1 shows the results of our evaluation. We see that the Bigram model is weak on both resources. The results of the String Goal model suggest that, even using the generous evaluation we do here, there

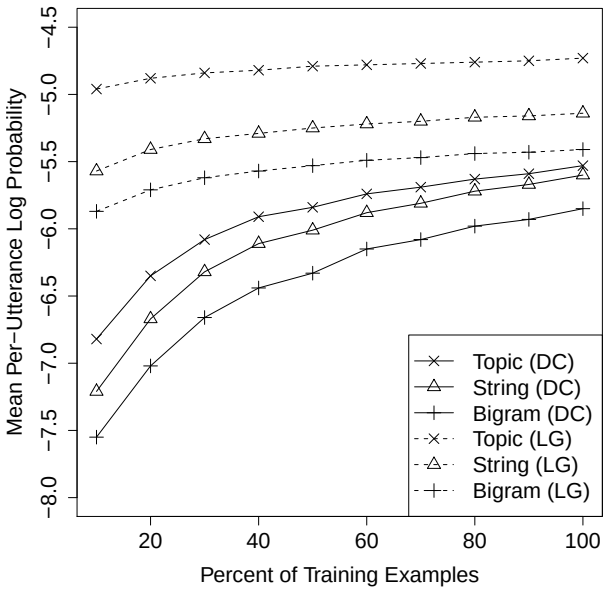

Figure 1: Heldout probability of the two resources for varying percentages of training dialogs. Note that while the percentages match across resources, Let's Go is much larger and thus the absolute numbers of dialogs are different, which explains the better performance on Let's Go.

is much variability due to synonymy and recognition errors which string goals are unable to capture (in contrast to our Topic Goal model). The Topic Goal model explains this much more easily by grouping commonly co–occurring values into the same topic. Table 3 shows the perplexities corresponding to the performances with 100% training data for all acts and just PROVIDE acts (perplexity is 2^{-lp} where lp is the log probability). Improvements are more apparent when we compute the probability over PROVIDE acts alone, which the models are designed to handle. And since perplexity is not on a log scale, the differences are more pronounced. The Act model, which is a bigram model over $\{$ACT, *slot*$\}$ pairs alone excluding the values, demonstrates the vast discrepancy in uncertainty between the full problem and the valueless prediction problem. We note that the perplexity of our Act model on Communicator is comparable to that of Georgila et al. (2006).

6.2 Example Simulator Behaviour

In this section we give examples of our Topic Goal model simulator in generation mode, which corresponds to sampling from the induced model.

d	z_{dest_city} [probability]	proportion of samples	user utterance given topic z_{dest_city} and machine utterance REQUEST_INFO *dest_city*
1	**Norfolk Virginia** [0.562]	0.264	PROVIDE_INFO *dest_city* **Norfolk Virginia**
	Norfolk [0.234]	0.111	PROVIDE_INFO *dest_city* **Norfolk**
	Newark Virginia [0.088]	0.039	PROVIDE_INFO *dest_city* **Newark Virginia**
	Virginia Beach [0.0412]	0.028	PROVIDE_INFO *orig_city* **Las Vegas Nevada**
	Newark [0.040]	0.028	NO_ANSWER *null* **no**
		0.025	COMMAND *start_over* **start over**
2	**Chicago** [0.350]	0.164	PROVIDE_INFO *dest_city* **Chicago**
	Chicago Illinois [0.182]	0.082	PROVIDE_INFO *dest_city* **Chicago Illinois**
	Duluth Minnesota [0.124]	0.057	PROVIDE_INFO *dest_city* **New Orleans**
	New Orleans [0.122]	0.055	PROVIDE_INFO *dest_city* **Duluth Minnesota**
	New Orleans Louisiana [0.085]	0.039	PROVIDE_INFO *dest_city* **New Orleans Louisiana**
		0.028	NO_ANSWER *null* **no**
3	**Anchorage** [0.539]	0.252	PROVIDE_INFO *dest_city* **Anchorage**
	Anchorage Alaska [0.148]	0.072	PROVIDE_INFO *dest_city* **Anchorage Alaska**
	Jacksonville Florida [0.124]	0.056	PROVIDE_INFO *dest_city* **Jacksonville Florida**
	Great Anchorage Alaska [0.098]	0.048	PROVIDE_INFO *dest_city* **Great Anchorage Alaska**
	Duluth Minnesota [0.057]	0.047	PROVIDE_INFO *orig_city* **Hartford Connecticut**
		0.026	PROVIDE_INFO *dest_city* **Duluth Minnesota**

Table 4: Examples of sampling from the topic goal model. Left: top 5 strings (with probabilities) sampled from topics for three different dialogs *d*. Right: top 6 utterances (plus fraction of samples in 10,000) generated in response to the machine utterance "REQUEST_INFO *dest_city*" and conditioned on the topic z_{dest_city}.

Our examples are drawn from the model induced for the Communicator data. Sampling from standard distributions can be implemented following the algorithms in Bishop (2006) and other statistical resources. Utterances are sampled by sampling ACT, *slot* pairs from the distribution $p\left(a_i, s_i | m_{i-1}\right)$ (drawing a value from a multinomial distribution). If we sample a PROVIDE_INFO act, we check whether we have sampled a topic for the corresponding slot thus far in the dialog. If not, we sample one by drawing a topic indicator from $p(z_s) = \frac{\gamma_{rz}}{\gamma_{\bullet\bullet}}$ and then drawing a multinomial distribution over strings from the Dirichlet posterior corresponding to z. Once the topic for the slot is set, we sample **values** as draws from the fixed multinomial and add these to the ACT, *slot* pair.

Table 4 shows some examples drawn from the model. For each row in the table (corresponding to a new dialog *d*), we sample a topic for the *dest_city* and *orig_city* as needed, and sample 10000 utterances given that topic. The left hand side of the table shows the top five strings in the sampled topic, while the right hand side shows the top six utterances in response to REQUEST_INFO *dest_city*. Note that the proportion of utterances on the right does not match the probability of the values on the left because of the presence of other user acts besides PROVIDE *dest_city*.

7 Evaluating Model Consistency

Having shown in the previous section that our Topic Goal model is a much better predictor of heldout data than the String Goal model or Bigram model, we now turn to a demonstration of the model's capturing of consistency.

In the face of value synonymy and ASR errors, we define *inconsistent* dialogs to be ones that are locally coherent but lack the structure of a real dialog from one turn to the next. We then suggest that an appropriate task for consistent models is distinguishing between consistent and inconsistent dialogs.

To test this hypothesis, we devise the following classification problem: can we discriminate between

Baseline	Dialog length (turns)
	Mean, standard deviation, min and max acts per turn
	Presence of special machine acts (flight offer and confirm)
	Presence of user acts (provide a dest city and arrival city)
	Proportion of acts which were provides
String Consistency	Did the user provide inconsistent information about dest city?
	Did the user provide inconsistent information about orig city?
Topic Model	Ranked list of posterior probabilities of top 50 topics
	Normalised probability of dialog for topic model

Table 5: Feature sets for consistency experiments

real dialogs and those generated by randomly sampling turns from different dialogs? In this section we induce classifiers over various feature sets to demonstrate that we can, and that the Topic Goal model contains far more useful information in this regard than string-based consistency features. (The bigram model by definition provides no help here, since the units of which dialogs consist contain the entire window of context used for the bigram model).

We take our training and development data from the Communicator corpus in the previous section, and create a classification problem as follows: real dialogs form positive examples in the classification problem. To create negative examples, we sample {machine, user} turns at random from the appropriate resource. We keep a histogram over real dialog lengths, and sample a number of turns for our "fake" dialogs proportional to this histogram. We then sample this many turns from the frequency distribution over turns in the real data, and create exactly as many dialogs in this fashion as real dialogs in the data. The result is an equal number of dialogs comprised of real turns, of (expected) real length, but where the sequence of turns is highly unlikely to be coherent given the random sampling. The classification problem is thus far from trivial. We do this from our training data to produce data with which to train the classifier, and from our development data to provide test instances. This gives rise to 2500 training instances, and 500 test instances.

We learn linear SVMs with various features described in Table 6. These feature sets are designed to capture different aspects of consistency: the baseline features are intended to capture surface level features of the dialogs, inspired by (Schatzmann et al., 2005) where they provide trivial separation of real from simulated dialogs. However, our setting is different: we do not seek to tell real dialogs from fully simulated ones, but real dialogs from scrambled versions of real dialogs. In addition to length-based features, we add binary presence indicator for several user and machine acts highly correlated with the completion of dialogs, as well as for acts which indicate the provision of information and the proportion of all acts occupied by these. The table gives a complete list of these Baseline (B) features.

We derive a second set of features intended to replicate the utility of string-based goals: we set up binary features to fire if contradictory information is provided for the slots over the course of the dialog. These are our String Consistency (SC) features.

Finally, we use our topic-model simulator to derive consistency features. Our features are the posterior distribution over topics for each slot given that dialog. Our topics are induced from the real training dialogs, and their posterior probabilities computed for all dialogs relative to this model. We take posterior probabilities of the fifty most probable topics for each of the *dest_city* and *orig_city* slots as features, as well as the normalised log probability of the dialog (the log probability divided by the number of user utterances). These form our Topic Model (TM) features.

Our classifiers are linear SVMs, and we use libsvm (Chang and Lin, 2011), scaling features to the range $[0 - 1]$. All other parameters are left at their defaults.

Feature Set	Accuracy
Baseline (B)	74.34 $\pm$3.77
String Consistency (SC)	63.60 $\pm$4.27
B + SC	77.63 $\pm$3.58
Topic Model (TM)	79.61 $\pm$3.44
B + SC + TM	85.96 $\pm$2.89

Table 6: Performances for the classifiers. Errors are 95% intervals to the accuracies assuming they are parameters to a binomial distribution

7.1 Results

The results of the classifiers are shown in Table 5. Since we have an equally balanced binary classification task, accuracy is the most appropriate metric. Here we see that the baseline and string consistency features have roughly the same discriminatory potential, and their union produces a slight improvement. The topic model features are far superior to this, and the union of all three sets gives a further improvement.

These results demonstrate that our model encodes notions of consistency which go substantially beyond those defined at the level of strings. Features defined over the latent topic goal space substantially improve performance in a difficult discrimination task, demonstrating that our model captures an important notion of how real dialogs appear that is not shared by the other models we consider.

8 Concluding Remarks and Future Work

This paper presents a fully generative goal driven user simulator, the first to merge both consistency and variability within a fully probabilistic framework. We evaluate our model on two task-based dialog domains, Let's Go and Communicator, and find it to outperform both a simple bigram model and an upper bound on probability models where the strings are represented as goals, in terms of the probability the model assigns to heldout dialogs.

We then move on to show that features derived from the model lead to substantial improvement in detecting real dialogs from those where the turns have been selected at random from all turns in the training data: this is a fairly difficult task, but our model allows significant improvement over strong and sensible baselines.

Our model could be extended in a number of ways. It could be improved to incorporate noise resulting from the decisions made by the semantic parser. Another possible improvement is to explore the effects of introducing dependency between the slots in the user goal, which would enforce more plausible values pairings and would potentially improve the simulator's performance. The effects of a dependence assumption between the different utterances occurring in a single user turn under the act model can also be explored. We would also like to use our simulator to train a POMDP-based dialog manager using a form of reinforcement learning.

References

Hua Ai and Diane J. Litman. 2008. Assessing dialog system user simulation evaluation measures using human judges. In *Proceedings of ACL-08: HLT*.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Alan W. Black and Maxine Eskenazi. 2009. The Spoken Dialogue Challenge. In *Proceedings of SIGDIAL 2009*, SIGDIAL '09.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*.

M. Gašić, F. Jurcicek, B. Thomson, K. Yu, and S. Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding, 2011 IEEE Workshop on*, Hawaii, December.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2005a. Learning user simulations for information state update dialogue systems. In *Proceedings InterSpeech 2005*.

Kallirroi Georgila, Oliver Lemon, and James Henderson. 2005b. Automatic annotation of communicator dialogue data for learning dialogue strategies and user simulations. In *Proceedings Ninth Workshop on the Semantics and Pragmatics of Dialogue*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User Simulation for Spoken Dialogue Systems:

Learning and Evaluation. In *Proceedings InterSpeech 2006*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *PNAS*, 101(suppl. 1):5228–5235.

Srinivasan Janarthanam and Oliver Lemon. 2009. A two-tier user simulation model for reinforcement learning of adaptive referring expression generation policies. In *Proceedings of SIGDIAL 2009*, SIGDIAL '09, pages 120–123.

Sangkeun Jung, Cheongjae Lee, Kyungduk Kim, Minwoo Jeong, and Gary Geunbae Lee. 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer Speech & Language*, 23(4):479–509.

Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of SIGDIAL 2010*.

Esther Levin and Roberto Pieraccini. 2000. A stochastic model of human-machine interaction for learning dialog strategies. In *IEEE Transactions on Speech and Audio Processing*.

E. Levin, S. Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Di Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. 2000. The AT&T-DARPA communicator mixed-initiative spoken dialog system. In *In ICSLP*.

Radford Neal. 1991. Bayesian Mixture Modeling by Monte Carlo Simulation. Technical report, University of Toronto.

Olivier Pietquin. 2004. *A Framework for Unsupervised Learning of Dialogue Strategies*. Ph.D. thesis, Faculté Polytechnique de Mons, TCTS Lab (Belgique), apr.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *Proceeings of 6th SIGDIAL Workshop*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007a. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *HLT-NAACL (Short Papers)*, NAACL-Short '07.

Jost Schatzmann, Blaise Thomson, and Steve Young. 2007b. Statistical User Simulation with a Hidden Agenda. In *Proceedings 8th SIDdial Workshop on Discourse and Dialogue*, September.

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT 2002*.

Satinder P. Singh, Michael J. Kearns, Diane J. Litman, and Marilyn A. Walker. 2000. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Jason D. Williams. 2008. Evaluating user simulations with the Cramer-von Mises divergence. *Speech Communication*, 50(10):829–846, October.